Utilization of Navigational Queries for Result Presentation and Caching in Search Engines

Rifat Ozcan, Ismail Sengor Altingovde, Özgür Ulusoy
Bilkent University, 06800, Bilkent, Ankara, Turkey
{rozcan, ismaila, oulusoy}@cs.bilkent.edu.tr

ABSTRACT
We propose result page models with varying granularities for navigational queries and show that this approach provides a better utilization of cache space and reduces bandwidth requirements.

Categories and Subject Descriptors
H.3.4 [**Information Storage and Retrieval**]: Systems and Software – *Performance evaluation (efficiency and effectiveness).*

General Terms
Design, Experimentation, Performance.

Keywords
Navigational queries, search engine, static caching.

1. INTRODUCTION
Web users' search goals are usually categorized as "informational" or "navigational" [3]. A navigational query is intended to find a particular Web site that a user has in mind. Therefore, the search process will probably end up with one (or two) click(s) for the *Top-1* (or *Top-2*) results from the first page of results. However, Web search engines (WSEs) usually cache and return the results of queries in a standard form of 10 results per page, regardless of the query type. For navigational queries, this may cause wasting cache space and network bandwidth. In this paper, we propose result page models with varying granularities for navigational queries. This approach is shown to improve bandwidth usage during result display in turn of a small increase in the number of result pages inspected by the users. Furthermore, a better space utilization is obtained for static result caching.

2. PAGE MODELS FOR RESULT DISPLAY
We first define some of the basic notions as follows.

Definition 1 *(Result Page Model):* A result page is an atomic item for internal (e.g., result caching) and external (e.g., result display) purposes of the WSE. A result page model describes how the query results are placed into the pages, each of which may include fixed or variable number of results.

Definition 2 (*numSnippetsSend):* This measures the number of snippets sent by the WSE to the users. It shows the network bandwidth cost incurred by the query result display.

Definition 3 (*numResultPagesBrowsed):* This indicates the total number of the result pages that the user will browse in order to reach the target document(s) for his/her query.

CIKM'08, October 26–30, 2008, Napa Valley, California, USA.
ACM 978-1-59593-991-3/08/10.

In the literature, it is reported that users rarely click on more than the top-20 results [5], so we restrict our analysis to the result page models for this most common case. Let's consider a query instance Q and the click requests $C= \{c_1, c_2, ..., c_k\}$ of k clicks for this query. Assume that click requests are at the ranks $R= \{r_1, r_2, ..., r_k\}$, where $r_k \leq 20$. Then the result at rank r_k is defined as the *lowest-ranked clicked document* for this query session. For simplicity, we assume that the user requested all the result pages until the result page containing the result at rank r_k and user will not request more results after this rank. We also ignore query sessions including no clicks. Based on these assumptions, we collect all <query_instance, lowest_ranked_clicked_document> pairs for top-20 clicks. Then, we end-up with a list $A = \{A_1,..., A_{10}, A_{11},..., A_{20}\}$ where A_i denotes the number of query instances for which the lowest-ranked clicked document is at rank i.

Assume we have a 2-page result model for top-20 as X_Y which denotes that the first result page contains X results and the second result page contains Y (or 20-X) results. We derive formulas for the cost measures as follows. Note that, the cost formulations can be generalized for top-K results and M pages in a straightforward manner, as we also consider 3-page result models in experiments.

$$numSnippetsSend_{X\_Y} = X * \sum_{i=1}^{20} A_i + Y * \sum_{i=X+1}^{20} A_i \qquad \textbf{(1)}$$

$$numResultPagesBrowsed_{X\_Y} = \sum_{i=1}^{20} A_i + \sum_{i=X+1}^{20} A_i \qquad \textbf{(2)}$$

We normalize these expressions using the conventional result model of 10-results-per-page schema (i.e., 10_10) as our baseline model. The summation of these normalized values is used as an overall measure for the evaluation of the result presentation models in the experiments below.

Dataset. We use a subset of the AOL Query Log (http://imdc.datcat.org/collection/1-003M-5). Our subset contains 4,276,944 query instances that have at least one click. Among those queries, 2,315,435 of them are submitted in the first 6 weeks and reserved as the train set. The train set is used to determine the navigational queries and fill the static cache (discussed in the next Section). The remaining 1,961,509 queries constitute the test set.

Identifying navigational queries. In this study, we adopt a simple and effective approach from [3] and slightly extend it for more flexibility. That is, if the click count for top-1 and top-2 results constitutes the 90% of all clicks for that query then it is classified as a navigational query. Additionally, we define the notion of confidence as in Equation 3 below:

$$C = (Freq\_of\_top2\_clicks) * (\log\_normalized\_query\_freq) \qquad \textbf{(3)}$$

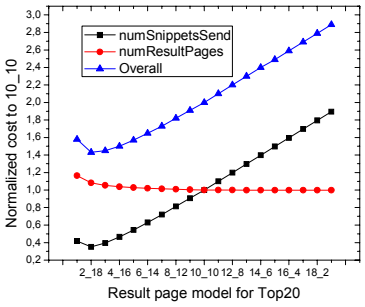

Figure 1. The costs of 2-page result presentation models.

In particular, we call those queries with top-2 click frequency greater than 80% and confidence score greater than 0.2 as navigational queries, as well. Finally, we exclude the queries which have only one click in the dataset or which occurs only once in the training log regardless of above considerations. In our train log, we discovered that among the 1,220,246 distinct queries, 89,442 of them are navigational queries.

Experiments. Figure 1 shows the graph for the normalized cost of 2-page result presentation models ranging from 1_19 to 19_1 for only navigational query types and top-20 clicks. The overall line in the graph shows that it is possible to improve the baseline approach. In this case, the best result presentation model is 2_18, which provides an overall improvement of 28.5%. We also conduct experiments with 3-page result presentation models. For only navigational queries, the best model is 2_8_10 which achieves an overall improvement of 32.5%.

3. PAGE MODELS IN STATIC CACHING

In a static cache of query results, either document identifiers (*docID cache*) or the actual HTML page (*snippet cache*, including snippets, etc.) can be stored [2]. For both cases, we examine the effect of using the result page model of 2_8_10, which is reported to achieve the best overall improvement in Section 2, for navigational queries. Typically, a static cache is populated with the most frequent query results from a previous query log (See [4] for other strategies). For the baseline case, query results consist of result pages of size 10 results. The static cache is filled with most frequent *<query, result_page_no>* items in the training log. In contrast, for our result page model, the size of the cached items must be taken into account in addition to frequency. The items are ordered by the following score formula (adapted from [1]).

$$S_{<query,result\_page\_no>} = \frac{FREQ_{<query,result\_page\_no>}}{SIZE_{<query,result\_page\_no>}} \quad (4)$$

Experiments. Since the result page sizes are not the same in our caching approach and in the baseline, using cache hit rate would not be a fair measure. Therefore, we use the *absolute* number of the cache misses to measure the effectiveness of two caching strategies. During the initial experiments we realized that for some of the queries that are identified as navigational in the train log and have all their clicks for the top-2 results, our scheme would never cache the second result page (with results 3 to 10) regardless of the cache size. That is, if all clicks are for the top-2 results, the frequency of second result page would be 0, and can never be cached. This may reduce the utility of our approach against the baseline for large cache sizes. To remedy this problem, we use the confidence score used during the identification of navigational queries also for a smoothing operation. That is, for each query identified as a navigational query, we multiply its frequency with (1- confidence)/*c*, where *c* is an experimental constant, and add this score to the frequency of the second result page. This creates a smoothing effect and allows us to cache the second result page of a navigational query with low confidence, even if it is never requested in the train log.

In Table 1, we report the experimental results with smoothing. The cache size is given as the number of 10-results-per-page cache size entries. The reduction percentages in the total miss counts are shown in a separate column. The reductions are higher in the snippet cache case since the space gain in that case is higher than the case of a docID cache. On the other hand, as the cache size increases, the improvements decrease since it causes many of the second result page of the navigational queries to be cached also. Anyway, the proposed result page model provides reductions of 1-2% and 3-4% in absolute miss counts for the docID and snippet caches, respectively. When compulsory misses (i.e., queries that occur only in the test log) are excluded, reductions are more emphasized, reaching up to 4% and 11.6% for the docID and snippet caches, respectively.

Table 1. Static caching performances with smoothing

Cache Size	Baseline	DocID cache (with 2_8_10)	% red	Snippet Cache	% red
5K	1,948K	1,923K	1.28	1,876K	3.69
10K	1,876K	1,855K	1.13	1,818K	3.12
20K	1,799K	1,784K	0.85	1,759K	2.25
50K	1,697K	1,685K	0.75	1,663K	2.01
100K	1,626K	1,617K	0.55	1,599K	1.68
200K	1,570K	1,563K	0.42	1,554K	0.98
500K	1,518K	≈1,517K	0.04	≈1,517K	0.07
1000K	1,465K	≈1,465K	0.03	≈1,465K	0.04

4. ACKNOWLEDGMENTS

This work is supported by The Scientific and Technological Research Council of Turkey (TÜBİTAK) by grant# 108E008 and 105E065.

5. REFERENCES

[1] Baeza-Yates, R., Gionis, A., Junqueira, F., Murdock, V., Plachouras, V., and Silvestri, F. The impact of caching on search engines. In *Proc. of SIGIR '07*, ACM, 2007, 183-190.

[2] Fagni, T., Perego, R., Silvestri, F., and Orlando, S. Boosting the performance of Web search engines: Caching and prefetching query results by exploiting historical usage data. *ACM TOIS 24*, 1 (Jan. 2006), 51-78.

[3] Lee, U., Liu, Z., and Cho, J. Automatic identification of user goals in Web search. In *Proc. WWW '05*, 2005, 391-400.

[4] Ozcan, R., Altingovde, I. S., and Ulusoy, Ö. Static query result caching revisited. In *WWW '08*, 2008, 1169-1170.

[5] Silverstein, C., Marais, H., Henzinger, M., and Moricz, M. Analysis of a very large web search engine query log. *SIGIR Forum 33,* 1 (Sep. 1999), 6-12.